Space-Time Tradeoffs for Parallel 3D Reconstruction Algorithms for Virus Structure Determination

Dan C. Marinescu, Yongchang Ji, and Robert E. Lynch
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907
Email: [dcm, ji, rel]@cs.purdue.edu

March 30, 2001

Contents

Abstract

The 3D electron density determination of viruses, from experimental data provided by electron microscopy, is a data intensive computation that requires the use of clusters of PCs or parallel computers. In this paper we report on three parallel algorithms used for 3D reconstruction of asymmetric objects from their 2D projections. We discuss their computational, communication, I/O, and space requirements and present some performance data. The algorithms are general and can be used for 3D reconstruction of asymmetric objects for applications other than structural biology.

1 Introduction

Very often data intensive computations demand computing resources, CPU cycles, main memory, and secondary storage, well beyond those available on sequential computers. To solve data intensive

problems using parallel systems, either clusters of PCs or high performance systems, we have to design families of parallel algorithms and select the most suitable one for the specific problem size, the architecture of the target system, the performance of its sub-systems, and the amount of resources available. For example, if a system has limited memory, a high latency and low bandwidth interconnection network, and very fast processors, then we might chose to replicate some computations in each node. Such space-time tradeoffs are common in the design of efficient parallel algorithms.

We are interested to select an optimal algorithm in a family of algorithms, together with its control parameters, for a particular problem and a given computing environment. This should be done automatically by the software after the program input has been specified. A scientist using the system need not be concerned with the complexity of the algorithms or the intricacies of using a parallel system.

Our primary interest is computational structural biology and the solution we propose is the use of agents to configure, monitor, and control the execution of a parallel program used for 3D electron density determination of viruses based upon experimental data from electron microscopy. The contributions of this paper include: three parallel algorithms for 3D reconstruction of asymmetric objects in Cartesian coordinates, an analysis of these algorithms and performance data for several virus structures.

The paper is organized as follows. Section 2 presents related work, Section 3 introduces the reader to the basic techniques for atomic structure determination of large macromolecules like viruses. Section 4 outlines the motivation for parallel algorithms and argues that increasing the resolution of the structure determination and solving structures with no symmetry represents a quantum leap in virus structure determination using experimental information about the virus structure, gathered from electron micrographs. In Section 5 we describe three parallel algorithms and present a succinct analysis of their computational and communication complexity as well as their space requirements. Then, in Section 6 we briefly discuss the quality of the solution obtained with programs implementing the algorithms and in Section 7 we discuss performance results. Finally we present our conclusions and discuss future work.

2   Related Work

The Radon transform [Rad17] provides an analytical solution to the problem of reconstructing a 3D object from a set of 2D projections. In late 1960's and early 1970's the interest in 3D reconstruction was renewed by the introduction of computerized tomography in medicine and by the electron microscopy, [DeR67]. The group of Aaron Klug at Cambridge, is responsible for analytical results including the "projection or central section theorem", as well as the experimental reconstruction of helical fibers [DeR70] and icosahedral viruses [Cro70].

There are several practical methods for reconstructing a 3D object from a set of its 2D projections. The methods for 3D reconstruction include use of Fourier Transforms, 'back projection', and numerical inversion of the Radon Transform. See [Gor74] for a review of these and other methods.Also, for descriptions of (sequential) methods for 3D reconstruction and related tasks, see [Dea93], [Fra96], and [Gra96], three of several books containing clear explanations and many references.

There are subtle differences between the applications of 3D reconstruction in computerized to-mography and in cryo-TEM for atomic virus structure determination. There are two basic ap-proaches used in electron microscopy: single-particle and multiple-particle techniques. The classical tomographic approach requires many views of the same particle tilted in various orientations. In

case of computer assisted tomography, a computer controls the angles and the entire data collection process. The tilts required for by these approaches limit the resolution level. The single-particle approach is described in [vHe00].

In this paper we only discuss 3D reconstruction using multiple particles. In this case the biological specimen consists of multiple copies of the virus, each frozen in a different orientation. Determining the orientation of each view is a major problem in this case. The method discussed in this paper is iterative, we start with an original electron density map, project it at known angles and use these projections to determine the orientation of each experimental "view". Once we have estimated the orientation of each experimental view we carry out the 3D reconstruction and hopefully obtain a better electron density map, that is used to determine better orientation, and the process continues until a stopping condition is met, e.g., the orientations no longer improve.

An important problem is the radiation sensitivity of the biological material. In electron microscopy the information about the specimen is provided by the electrons that have interacted elastically with the specimen. An intense beam of electrons will damage the specimen by non-elastic interactions. As a result, the signal-to-noise ratio is very low and the quality of the micrograph is very poor.

Last but not least, the Contrast Transfer Function, see Section 6, plays a critical role, in cryo-TEM 3D reconstruction as discussed in depth in [vHe00].

The basic ideas of the maximum likelihood methods for 3D reconstruction is that the best model is the most consistent with the observations. The consistency is measured by the probability that the observation would be made given the current model. When the model is modified to make the observations more probable, the model gets better and the likelihood increases. The probability that the current model leads to the actual observations includes the effects of all sources of errors and as the model improves the errors decrease. Maximum likelihood methods for protein structure determination are discussed in [Pan96], [Mur97], and [Bri96].

A recent paper covers the subject of ab initio reconstruction for cryo-TEM [Doe00]. The paper introduces a statistical model and uses maximum likelihood reconstruction. The 3D structure of several viruses is computed using an expectation maximization algorithm.

To use efficiently a parallel computer or a cluster of workstations, we need parallel algorithms that partition the data and computations evenly among nodes to ensure load balance and, moreover, which minimize the communication among processors by maintaining a high level of locality of reference. Similar efforts have been reported in the past [Joh94], but the performance data available to us suggest that new algorithms have to be designed to reduce dramatically the computation time and to support reconstruction of asymmetric particles.

3 3D Reconstruction in Electron Microscopy

X-ray crystallography, electron microscopy, EM, and Nuclear Magnetic Resonance, NMR, are techniques used for gathering experimental information about the 3D atomic structure of biological material. Viruses are large macromolecules with hundreds of thousands of amino-acids and millions of atoms. The atomic structure of viruses is of considerable interest for the design of antiviral drugs.

The 3D atomic structure determination of macromolecules based upon electron microscopy is an important application of 3D reconstruction. The procedure for structure determination consists of the following steps:

Step A Extract individual particle projections from micrographs and identify the center of each
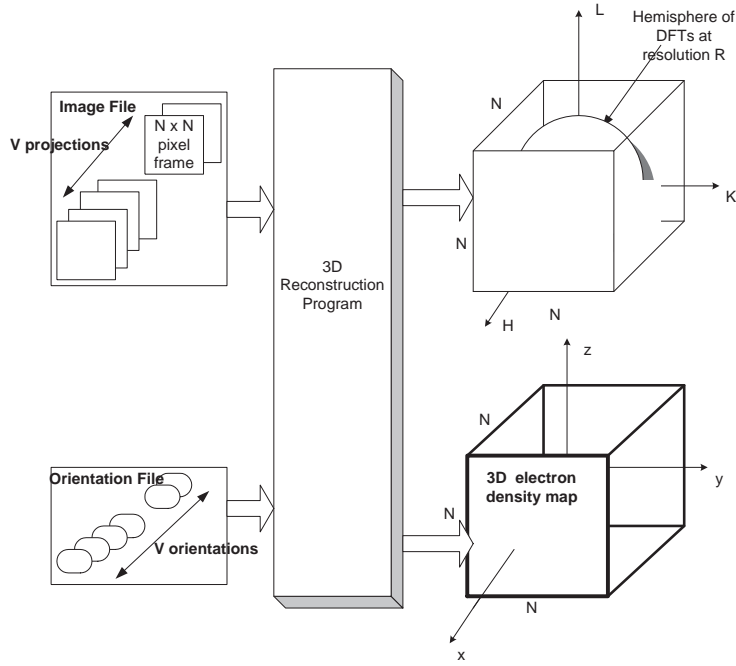
Figure 1: The **3D** reconstruction. The input to the program are: a file with V, $N \times N$ pixel images, each containing a projection of the virus, and one file with the orientation of each projection. The reconstruction is done in the **3D DFT** domain and is limited to a hemisphere of radius R. The radius R is determined by the desired resolution of the reconstruction. Finally the **3D**, $N \times N \times N$ electron density map is obtained through an inverse **3D DFT**.

projection.

Step B Determine the orientation of each projection.

Step C Carry out the **3D** reconstruction of the electron density of the macromolecule.

Step D Dock an atomic model into the **3D** density map.

Steps B and C are executed iteratively until the **3D** electron density map cannot be further improved at a given resolution. The number of iterations for **3D** reconstruction is in the 10–20 range and one step of **3D** reconstruction for a medium size virus may take several hours on a sequential computer. It typically takes weeks or even months to obtain an electron density map using sequential programs for the orientation determination and for the **3D** reconstruction. The development of parallel algorithms to carry out some of these computations is part of an ambitious effort to design an environment for 'real-time electron microscopy', where results can be obtained in hours or days rather than in weeks or months. Algorithms for Step A, which include automatic identification of particle projections, the determination of the center and orientation of each virus particle projection are discussed elsewhere [Mar97], parallel algorithms to determine the orientation, Step B are presented in [Bak97].

In this paper we are only concerned with Step C of the process outlined above. We use a Fourier method based on the 'projection or central section theorem': The **3D** Fourier Transform $\mathcal{F}(h, k, \ell)$, of a function $f(x, y, z)$, evaluated at points of a plane containing the origin in (h, k, ℓ)-space is the same as the **2D** Fourier Transform of the projection $\int \int f$ of f onto the same plane in (x, y, z)-space.
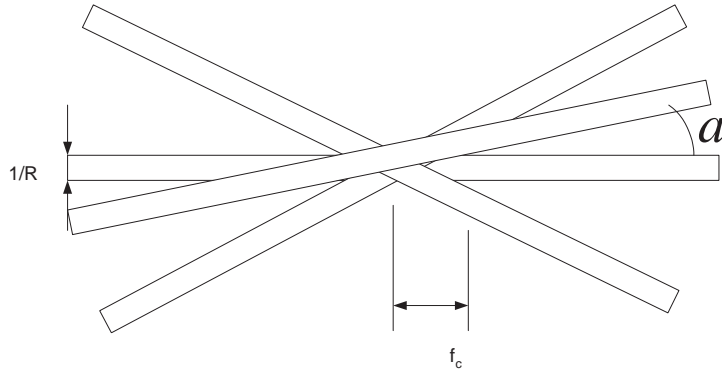
4

Figure 2: Central sections in the reciprocal space overlap at low frequencies. Two central section separated by an angle α overlap up to frequency f_c, see [vHe00].

We use the Discrete Fourier Transform (DFT) to approximate the Fourier Transform. The process is carried out as follows:

Step I Compute the 2D DFT of each projection.

Step II Use each 2D DFT and its orientation to estimate, by interpolation, values of the 3D DFT at points within half a grid spacing from the plane of projection. At each (h, k, ℓ), average the interpolated values obtained from the projections to obtain the final estimate of the 3D DFT.

Step III Compute the inverse 3D transform to get the electron density.

A 3D object has finite dimensions, R thus the 2D projection operation should only be carried out only up to distance R from the center. It follows that the central section is no longer an infinitely "thin" plane but a "slab" with thickness $\frac{1}{R}$, [vHe00]. Thus central sections overlap not only on a single line and the closer to the origin of the reciprocal space, the more the central sections used for reconstruction overlap as shown in Figure 2. This implies that low frequency components receive too much weight in the reconstruction and the 3D volume will be "blurred". Thus we need a high-pass filtering of the projections. van Heel warns that algorithms filling in missing information by maximum entropy or by POCS, Projections Onto Convex Sets may lead to incorrect results, [Har86].

The basic algorithm for the reconstruction of asymmetric objects in Cartesian coordinates is presented in [Lyn97] and [Lyn99]. Figure 3 illustrates the basic idea of the algorithm.

An improvement of the algorithm that reduces the number of arithmetic operations by two orders of magnitude is outlined in [Lyn00], and a detailed presentation and analysis of the algorithms together with preliminary experimental results are given in [Lyn00a].

The algorithm for 3D reconstruction was designed to work well on workstation clusters. The program uses MPI (Message-Passing Interface) and consists of several phases including initialization, 2D Fourier analysis, estimating the 3D coefficients of the DFT, interpolation, and Fourier synthesis.
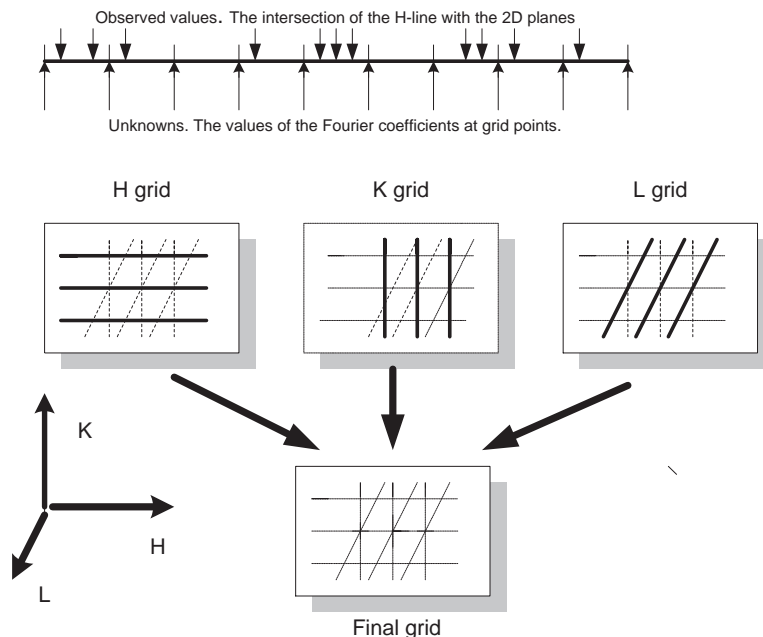
Figure 3: One-dimensional interpolation on a line parallel with the axes of the 3D grid in the Fourier domain. The values of the function at grid points (arrows pointing upwards) are unknown, the ones corresponding to arrows pointing downwards are the known, they are the observed values. We can determine by interpolation the values at grid points for all h-lines and construct an H-grid. Similarly we obtain the K-grid and the L-grid.

4 Motivations for Parallel Algorithms for Reconstruction of Asymmetric Objects

One of the 3D reconstruction algorithm proposed by Crowther, DeRosier, and Klug [Cro70] over 30 years ago has been used extensively by the structural biology community. The algorithm is based upon Fourier-Bessel Transforms and can be used for reconstruction of symmetric objects.

The protein shell of spherical viruses exhibit various degrees of symmetry, but the core of the virus, consisting of genetic material, does not. Structural studies of the virus core provide one of the motivations for 3D reconstruction algorithms of asymmetric objects. The amount of experimental data for the reconstruction of an asymmetric object is considerably larger than that necessary for a symmetric one. While a typical reconstruction the shell of an icosahedral virus at, say, 20Å resolution may require a few hundreds projections, e.g., 300, the reconstruction of an asymmetric object of the same size and at the same resolution would require 60 times more data, e.g., $18,000$ projections. See [Ros98] for the minimal number of projections as a function of the desired resolution and the size and symmetry of the virus.

X-ray crystallography has been the only method to obtain high resolution (2-2.5Å) electron density maps for large macromolecules like viruses, while until recently electron microscopy was only able to provide low resolution (20Å) maps. Cryo-EM is appealing to structural biologists because crystallizing a virus is sometimes impossible and even when possible, it is technically more difficult than preparing samples for microscopy. Thus the desire to increase the resolution of Cryo-EM methods to the 5Å range. Recently, successful results in the 7–7.5Å range have been reported,

[Böt97], [Con97]. But increasing the resolution of the 3D reconstruction process requires more experimental data. It is estimated that the number of views to obtain high resolution electron density maps from Cryo-EM micrographs should increase by two order of magnitude from the current hundreds to tens of thousands.

The amount of experimental data further increases when structural studies of even larger virus-antibody complexes are attempted. The use of larger zero fill ratios to improve the accuracy of 3D reconstruction [Lyn00a] increases the number of arithmetic operations and the amount of space needed for reconstruction.

Thus it is not unrealistic to expect an increase in the volume of experimental data for high resolution asymmetric objects by three to four orders of magnitude in the near future. Even though nowadays faster processors and larger amounts of primary and secondary storage are available at a relatively low cost, the 3D reconstruction of asymmetric objects at high resolution requires computing resources, CPU cycles, and primary and secondary storage, well beyond those provided by a single system. Thus the need for parallel algorithms.

5 Basic Concepts: Algorithms and Notations

The design of a parallel algorithm requires space-time tradeoffs. In this section, we present three algorithms for reconstruction which have different space, communication, I/O, and computational requirements. The three algorithms apply to the case that the parallel system does not support parallel I/O operations.

When a system with a parallel file system is used, then each processor can read directly that segment of the input data it needs, and, similarly, write that segment of the output data it produces. In contrast, with a system which does not have a parallel file system, then the reading of data and writing of output by each processor leads to a larger execution time because of I/O contention.

The clusters of PCs and the SP2 system which we have been using do not have parallel file systems. To avoid the increased time which results from I/O contention, we use a Coordinator-Worker paradigm. The coordinator (a processor), C, reads the the input images and orientations and then distributes them to the workers, $W_i, 1 \leq i \leq P - 1$; C uses gather operations to collect the final result.

The three algorithms outlined below use different strategies for distributing the input data to the workers and they use different data structures. These differences effect the amount of storage required and the execution time; experimental results are given in Section 7.

In addition to using (x, y, z) to denote a grid point in 'real space' and (h, k, ℓ) to denote a grid point in 'reciprocal space', we use the following notations:

$N \times N$: the number of pixels in one projection.

$K, \quad K \geq 1$: zero fill aspect ratio; the $N \times N$ pixel array is put into a $KN \times KN$ array with zero fill before the DFT is computed.

P : the number of processors.

C : the coordinator, the processor performing the I/O operations.

$W_i, \quad 1 \leq i \leq P - 1$: the processors that carry out tasks assigned to them by the Coordinator.

V : the number of projections.

$v = \lceil V/P \rceil$: number of projections processed by W_i in Algorithms I and III.

$v' = V - (P-1)v$: the number of projections processed by C in Algorithms I and III.

7

q' : the number of projections processed by one processor in Step 2 of Algorithm III.

$q = Pq'$: the number of projections read in one I/O operation in Algorithm III.

$q_{stages} = V/q$: the number of pipeline stages in Algorithm III.

p : the edge length, in Å, of a pixel; grid spacing in real space.

$p' = 1/(Kp)$: the grid spacing in reciprocal space.

r : the desired resolution in Å.

R : number of grid spaces in reciprocal space to obtain the desired resolution.

n : the order of the symmetry group; one projection can be used n times with different orientations.

h-slab : complex DFT values, $\mathcal{F}(h, k, \ell)$, for $(KN)^2$ grid points (k, ℓ) for each of $K\,N/2P$ grid lines h.

z-slab : complex DFT values, $\mathtt{F}(h, y, z)$, for $K\,N^2$ grid points (h, y) for each of N/P grid lines z.

Algorithm I. In Algorithm I, there are no replicated computations and there is no overlapping of communication with computations. Each processor must have space to hold the entire complex 3D DFT $\mathcal{F}(h, k, \ell)$. The amount of space required is $(KN)^3/2$ thus the algorithm is space-intensive $(KN)^3/2$.
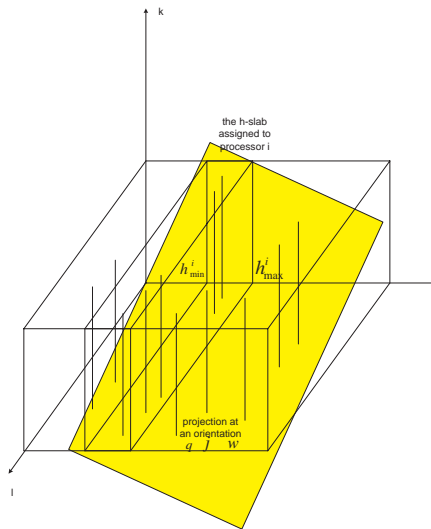

Figure 4: Filling up the grid in reciprocal space for Algorithm I. Each processor allocates storage for the entire 3D volume in the reciprocal space and uses the information provided by a subset v of projections to compute values at grid points scattered throughout the entire volume. It does so by intersecting each projection with the h, k, and ℓ lines (the k lines are illustrated). Then it retains the values in the h-slab allocated to it, scatters the values it has calculated in other h-slabs, and gathers the values it needs to completely fill out its own slab.

The algorithm consists of the followings steps:

1. C reads all V projections, keeps v' projections, and scatters a different group of v projections to each processor W_i, $1 \leq i \leq P - 1$.

2. C reads the orientation file and scatters a group of v orientations to the W_i, $1 \leq i \leq P - 1$, so that each processor has the orientation of each projection assigned to it.

3. After zero fill, each processor performs a 2D DFT for each of the v (or v') projections assigned to it.

4. Using the 2D DFT of each projection assigned to it, each processor obtains, by interpolation, estimates of the 3D DFT of the electron density at grid points (h, k, ℓ) near the plane of projection and inside the hemisphere of radius R; see Figure 4.

5. If the object has n-fold symmetry, then the interpolation of Step 4 is repeated for each of the other $n - 1$ symmetrically related orientations of the 2D DFT of each projection.

6. The P processors perform a global exchange. At the end, the i-th processor has its h-slab of interpolated values, as computed by it and by each of the other the processors. It averages these values and obtains the final estimate of the 3D DFT at all grid points in its h-slab and inside the hemisphere of radius R.

7. The i-th processor performs KN/P inverse 2D DFTs in its h-slab: $\mathcal{F}(h, k, \ell) \rightarrow \mathbf{F}(h, y, z)$,

8. The P processors perform a global exchange and each ends up with its z-slab.

9. Each processor performs R^2/P 1D DFTs to obtain the final estimate of the electron density in its z-slab: $\mathbf{F}(h, y, z) \rightarrow f(x, y, z)$.

10. Each W_i, $1 \leq i \leq P - 1$, sends its slab of electron density to C, which writes it onto the output file.

The computation complexity for each processor by Algorithm I is:
$O(\lceil V/P \rceil (K N)^2 \log(K N) + n \lceil V/P \rceil R^2 + KNR^2 + (K N)^3 \log(K N)/P)$

The space required for each processor is: $O(\lceil V/P \rceil (K N)^2 + (K N)^3)$

Algorithm I performs two *scatter* operations at the beginning, followed by two *scatter-gather* operation and one *gather* operation at the end. The communications complexity of Algorithm I is:
$O(V N^2 + (K N)^3 P)$

Algorithm II. This algorithm attempts to minimize the amount of space. Each processor calculates the 2D DFT of each projection. The algorithm is computation intensive, computations are duplicated to reduce space and communication requirements. There is no overlapping of communication with computations.

1. C reads the orientation file.

2. C reads one projection at a time and broadcasts it to W_i, $1 \leq i \leq P - 1$.

3. C broadcasts the values of the orientation so that each W_i, $1 \leq i \leq P - 1$, has the orientation of the projection assigned to it.

4. After zero fill, each processor performs a 2D DFT for the projection.

5. Each processor calculates the 3D DFTs at a subset of grid points, using the information provided by each projection, inside the h-slab allocated to it, see Figure 5.
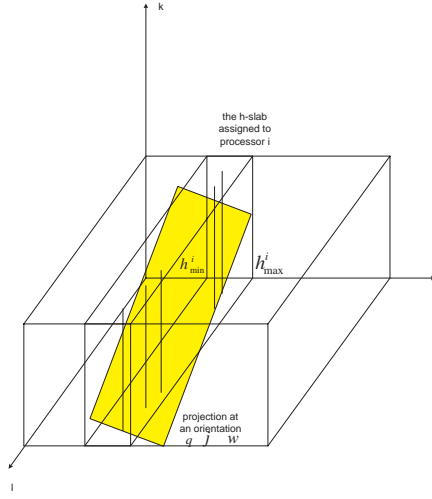
Figure 5: Filling up the grid in reciprocal space for Algorithm II. Each processor maintains a copy of the h-slab allocated to it and uses the information provided by all V projections to fill out this slab. It does so by intersecting each projection with the h, k, and ℓ lines in its h-slab (the k lines are illustrated).

6. If the object exhibits any type of symmetry, Step 5 is repeated for all the $n-1$ other projections related by symmetry to the one obtained experimentally.

7. Steps 2 to 6 are repeated until all projections have been processed. At the end of these steps, each processor has all the values inside the h-slab allocated to it.

8. The i-th processor performs KN/P inverse 2D DFTs in its h-slab: $\mathcal{F}(h, k, \ell) \rightarrow \mathbf{F}(h, y, z)$,

9. The P processors perform a global exchange and each ends up with its z-slab.

10. Each processor performs R^2/P 1D DFTs to obtain the final estimate of the electron density in its z-slab: $\mathbf{F}(h, y, z) \rightarrow f(x, y, z)$.

11. Each W_i, $1 \leq i \leq P-1$, sends its slab of electron density to C, which writes it onto the output file. h-slab, in

The computation complexity of algorithm II is:
$O(\ V\ (K\,N\ )^2\ \log(K\,N)\ +\ n\,V\,R\,K\,N/P\ +\ (K\,N)^3 \log(K\,N)/P\ )$

The space required for each processor is: $O(\ (K\,N)^2 + (K\,N)^3/P\ )$

Algorithm II performs two *broadcast* operations at the beginning, followed by one *scatter-gather* operation and one *gather* operation at the end. The communications complexity of Algorithm II is: $O(\ P\,V\,N^2 + (K\,N)^3\ )$

Algorithm III. This pipelined algorithm reduces the amount of space and allows overlapping of I/O and computations. No computations are duplicated.
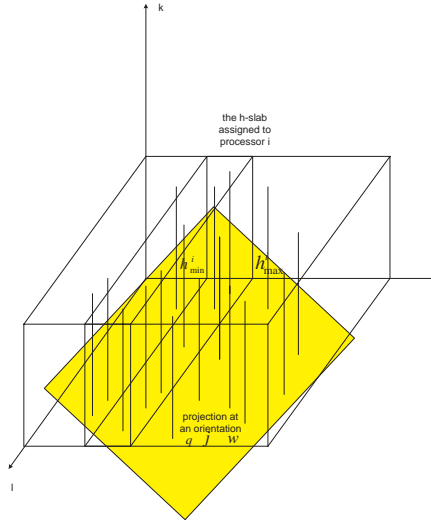
1. C reads the orientation file.

Figure 6: Filling up the grid in reciprocal space for Algorithm III. Each processor uses the information provided by a subset of projections to compute grid points scattered throughout the entire 3D volume. It does so by intersecting each projection with the h, k, ℓ lines (k lines are illustrated). It records the tuple $\{h, k, \ell, \mathcal{F}(h,k,\ell)\}$. Then it sorts the values and retains the values in the h-slab allocated to it, sends to all other processors the values it has calculated in other h-slabs, and gets from all the other processors the values it needs to completely fill out its own slab.

2. C reads groups of q projections at a time and scatters them in groups of size q' to W_i, $1 \leq i \leq P - 1$.

3. C scatters the values of the orientations so that each W_i, $1 \leq i \leq P - 1$, has the orientation of each projection assigned to it.

4. After zero fill, each processor performs a 2D DFT for each of the v projections assigned to it.

5. Each processor calculates the 3D DFTs at a subset of grid points using the information provided by each projection in the subset of v projections assigned to it, inside the entire 3D grid, see Figure 6. By interpolation it obtains all $\mathcal{F}(h,k,\ell)$ with (h,k,ℓ) within half a grid spacing from the projection. It stores the $\{h, k, \ell, \mathcal{F}(h,k,\ell)\}$ 4-tuples.

6. If the object exhibits n-fold symmetry, Step 5 is repeated, using the 2D DFT of Step 4, and each of the $n - 1$ symmetrically related orientations.

7. Each processor retains those $\{h, k, \ell, \mathcal{F}(h,k,\ell)\}$ 4-tuples within the h-slab allocated to it and exchanges each of the others with the the corresponding processors.

8. Steps 2 to 7 are repeated until all input projections are processed. At that time, at the end of Step 7, each processor has all the values of the 3D DFT at the grid points inside its h-slab.

9. The i-th processor performs KN/P inverse 2D DFTs in its h-slab: $\mathcal{F}(h,k,\ell) \rightarrow \mathbf{F}(h,y,z)$,

10. The P processors perform a global exchange and each ends up with its z-slab.

11. Each processor performs R^2/P 1D DFTs to obtain the final estimate of the electron density in its z-slab: $\mathbf{F}(h,y,z) \rightarrow f(x,y,z)$.
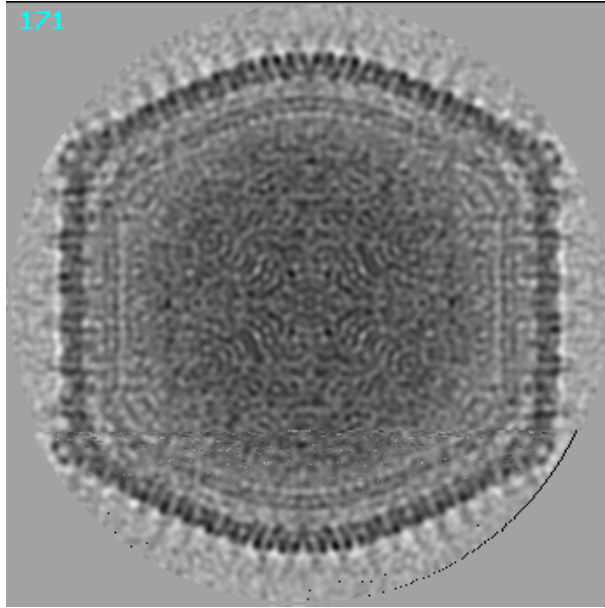
11

Figure 7: Cross section 171 of the CIV, Chilo Iridiscent Virus at 27Å resolution (Problem D). The reconstruction is done with the parallel program based upon the Algorithm I, running on an SP2 system. The time to produce the electron density map is 280 seconds using 32 processors. The size of a frame is 343×343 pixels. The zero fill aspect ratio is $K = 1.5$. The image for $K = 1.5$ is slightly better than that for $K = 1.0$.

12. Each W_i, $1 \le i \le P - 1$, sends its slab of electron density to C, which writes it onto the output file. h-slab,

The computation complexity of Algorithm III is:
$$O(\ \lceil V/P \rceil\ (K\,N\,)^2\ \log(K\,N)\ +\ n\ \lceil V/P \rceil\ (R^2 + (K\,N)^2)\ +\ (K\,N)^3\ \log(K\,N)/P\ )$$
The space required for each processor is: $O(\ n\,q'\,(K\,N)^2 + (K\,N)^3/P\ )$

Algorithm III performs two *scatter* operations at the beginning, followed by $q_{stages} + 1$ *scatter-gather* operations and one *gather* operation at the end. Recall that q_{stages} is the number of pipeline stages. The communications complexity of Algorithm III is: $O(\ V\,(K\,N)^2\ +\ (K\,N)^3\ )$

6 The Quality of the Solution

To assess the quality of the results produced with the parallel program we compared several electron density maps produced by our program with the ones produced by the sequential program [Bak98]

In Figures 7 and 8 we show a central cross section, section 171, of the electron density maps obtained with the parallel and with the sequential program, respectively. The result in Figure 8 is from [Yan00]. The virus is a lipid-containing, dsDNA icosahedral virus, the Chilo Iridescent Virus, CIV, with a diameter of 1850Å. The virus has a layered structure consisting of a dsDNA-protein core, surrounded by a lipid bilayer and icosahedral capsid shells consisting of thousands of sub-units. The outer diameters of the virus capsid range from 1615Å along the two and three-fold axes to 1850Å along the five-fold axis [Yan00]
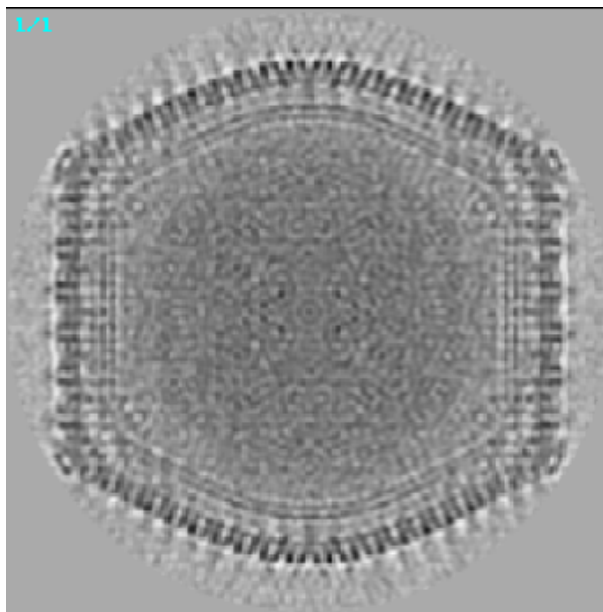
12

Figure 8: Cross section 171 of the CIV, Chilo Iridiscent Virus at 27Å resolution (Problem D). The 3D reconstruction is done with the sequential program, [Bak98], based upon an 3D algorithm due to Crowther, DeRosier and Klug. The size of a frame is 343×343 pixels and it is zero filled to 512×512. The image is from [Yan00]. The time to produce the electron density map is more than two hours using a 400 Mhz Alpha processor with 500 MBytes of memory.

The icosahedral protein shell is the object of current structural studies. An important problem for drug design is the binding of antiviral compounds to a virus and solving this problem requires the knowledge of the 3D structure of the protein shell. As one can see from Figures 7 and 8 the level of details of the protein shell, produced by the two reconstruction algorithms, are very similar. Icoahedral symmetry was imposed by the programs which produced Figures 7 and 8, so neither program produces an acurate representation of the non-symmetric core.

The relationship between the electron image of a specimen and the specimen itself is described by the Contrast Transfer Function, CTF, [Bak98]. The phase and the amplitude of the CTF are characteristic to a particular microscope, the specimen, and the conditions of imaging.

The CTF that enables the visualization of unstained specimens must be compensated in the reconstruction in order to achieve a reliable representation of the structure. The transfer function reverses, removes, and attenuates data in the image. The effects of the transfer function are more important at higher resolution. One can correct the transfer function by Winner filtering.

Our algorithm allows an effective CTF correction. In the interpolation phase we weight experimental data points coming from different micrographs with the corresponding value of CTF.

Figures 9 and 10 show cross-section 80-83 of $\phi X 174$, a bacteriophage virus with icosahedral symmetry at 10 Å resolution without and with CTF correction. The improvement in the quality of the reconstruction is visible.
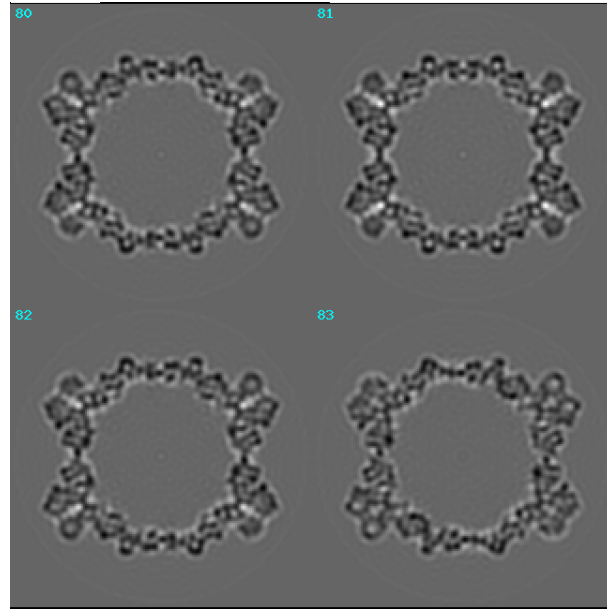
13

Figure 9: Crosssections 80-83 of $\phi X 174$, a bacteriophage virus with icosahedral symmetry, at 10 Å resolution. No CTF correction.
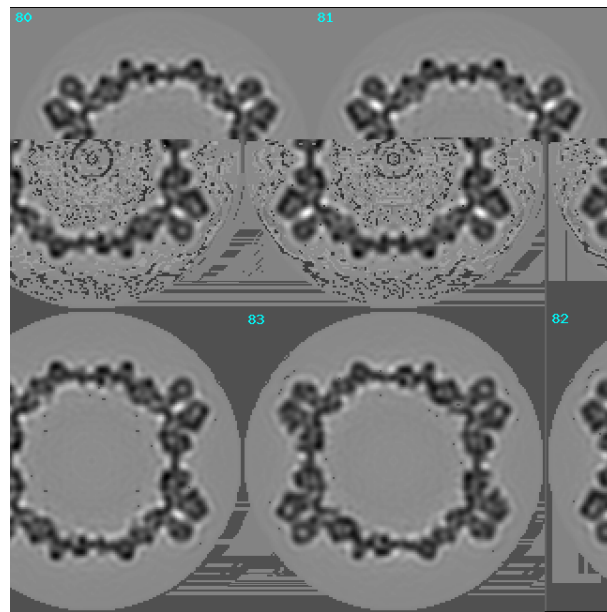

Figure 10: Crosssections 80-83 of $\phi X 174$, a bacteriophage virus with icosahedral symmetry, at 10 Å resolution. The CTF correction is applied.

Problem	Virus	Pixels	Views	Symmetry
A	Polyomavirus (Papovaviruses)	99 × 99	60 × 60	Icosahedral
B	Paramecium Bursaria Chlorella Virus, type 1	281 × 281	107 × 60	Icosahedral
C	Auravirus (Alphaviruses)	331 × 331	1940 × 60	Icosahedral
D	CIV Chilo Iridiscent Virus	343 × 343	667 × 60	Icosahedral
E	Herpes Virus	511 × 511	173 × 60	Icosahedral
F	Ross River Virus (Alphaviruses)	131 × 131	1777 × 10	Dihedral
G	Paramecium Bursaria Chlorella Virus, type 1	359 × 359	948 × 60	Icosahedral
H	Bacteriophage Phi29	191 × 191	609 × 10	Dihedral
I	Sindbis virus (Alphaviruses)	221 × 221	389 × 60	Icosahedral
J	Sindbis virus (Alphaviruses)	221 × 221	643 × 60	Icosahedral
K	Polyomavirus (Papovaviruses)	69 × 69	158 × 60	Icosahedral

Table 1: The 11 problems used to test the parallel 3D reconstruction program on an IBM SP2. The virus type, the number of pixels, the number of views/projections and the types of symmetry are indicated.

7 Performance Data

In the past we tested our algorithms on clusters of PCs, [Lyn00, Lyn00a]. Recently we implemented a version of the three algorithms for a high performance parallel system, an IBM SP2 system with 64 nodes, each having four-processors. The processors are POWER III, running at 375 Mhz. Each node has 4 GBytes of main storage and has a 36.4 GByte disk. The 4 processors in each node share the node's main memory and communicate, using MPI, with local processors and with processors in a different node.

The eleven sets of experimental virus data is listed in Table 1. Each of these viruses exhibit either icosahedral or dihedral symmetry. Our goal was to check the quality of our results compared with the ones obtained with sequential programs and, at the same time, to gather performance data. We measured the execution time and we recorded the memory usage when running in 1,2,4, 8, 16, and 32 processors. For each algorithm we also profiled the code to determine the time spent in each phase of execution when using a single processor. Tables 2, 3, and 4 list the results for Algorithm I. Tables 5, 6, and 7 list the results for Algorithm II, and Tables 8, 9, and 10 list the results for Algorithm III. Figures 11, 12, and 13 illustrate the speedups for the three algorithms. Table 11 presents ratio of the execution time and memory for Algorithms II and III versus Algorithm I. The execution times reported in Tables 2, 5, and 8, do not include the time to write the 3D electron density onto the output file.

The memory requirements of Algorithm I made it impossible to run Problems C, D, and G with less than two processors nor to run Problem E with less than 8 processors. The most computational intensive phases of Algorithm I are the 2D FFT analysis, the 3D FFT synthesis, and the 2D interpolation. The amount of memory used varied only slightly when the number of nodes increased, the largest problem, Problem E required slightly less than 700 MBytes of memory per node while problem A required less than 5 Mbytes on each of the 32 processors. The speedups for Algorithm I ranged from a low of 13.7 for Problem A to around 24 for Problems B, E, and J for 32 processors. The small size of Problems A and K limited their speedups.

Algorithm II is considerably more frugal in terms of memory than Algorithm I. Moreover, the memory required scaled down almost linearly when the number of processors increased, e.g., for

Processors	A	B	C	D	E	F	G	H	I	J	K
1	16.4	1126.1	–	–	–	437.2	–	470.9	519.5	812.7	28.0
2	8.4	566.4	3553.7	972.7	–	223.4	2335.0	241.0	262.8	413.6	14.1
4	4.5	287.6	1801.2	498.7	–	115.8	1186.3	126.3	136.0	210.5	7.4
8	2.6	149.6	968.2	265.4	376.9	64.2	617.7	72.1	71.0	110.0	3.9
16	1.6	78.9	493.3	146.8	266.4	38.3	337.5	39.7	39.5	59.7	2.2
32	1.2	46.6	275.9	84.2	132.5	21.8	186.7	22.3	23.7	35.1	1.5

Table 2: Algorithm I execution time (seconds).

Processors	A	B	C	D	E	F	G	H	I	J	K
1	11.6	266.3	–	–	–	27.2	–	84.1	129.5	129.5	3.9
2	5.8	133.6	217.8	242.4	–	27.2	277.9	84.1	65.1	65.1	2.0
4	4.9	111.2	181.4	201.9	–	27.6	231.5	84.1	54.1	54.1	1.7
8	4.9	110.9	181.3	201.8	667.2	28.4	231.3	84.1	54.0	54.0	1.6
16	4.9	110.9	181.3	201.8	667.2	30.1	231.3	84.1	54.0	54.0	1.6
32	4.9	110.9	181.3	201.8	667.2	33.4	231.3	84.1	54.0	54.0	1.6

Table 3: Algorithm I memory requirements (Mbytes).

Phase	A	B	F	H	I	J	K
FFT analysis	3.4	85.0	211.9	131.0	175.7	290.2	4.9
2D interpolation	9.4	97.1	178.8	134.9	273.4	452.1	21.4
Exchange	0.1	1.2	0.5	1.6	0.6	0.6	0.1
Averaging	0.1	3.6	0.5	1.6	1.7	1.7	0.1
2D Fourier synthesis	2.1	614.9	29.9	133.2	44.2	44.2	1.0
Exchange for fftsynth2	0.0	1.2	0.1	0.4	0.6	0.6	0.0
1D Fourier synthesis	1.0	309.1	14.9	66.8	21.2	21.2	0.5
Exchange for output	0.2	4.0	0.4	1.2	1.9	1.9	0.0
Writing output	6.3	24.5	3.7	7.7	12.6	15.2	1.3

Table 4: Time (seconds) for each phase of Algorithm I, one processor.

Processors	A	B	C	D	E	F	G	H	I	J	K
1	13.6	1089.7	6595.7	1744.0	2544.1	427.9	4361.0	438.0	450.2	700.8	21.6
2	10.0	604.0	5027.6	1372.2	1496.5	349.5	2734.6	303.4	354.4	563.8	14.8
4	7.7	357.0	4178.1	1164.9	964.7	298.3	1893.2	231.5	293.1	473.4	11.0
8	6.4	230.7	3744.6	1050.8	686.4	268.0	1454.4	216.1	254.4	416.1	9.1
16	5.7	166.8	3524.2	984.3	543.3	254.2	1221.4	176.2	233.6	383.3	8.3
32	5.1	137.0	3416.6	951.1	468.2	247.1	1101.6	166.1	224.6	368.7	7.7

Table 5: Algorithm II execution time (seconds).
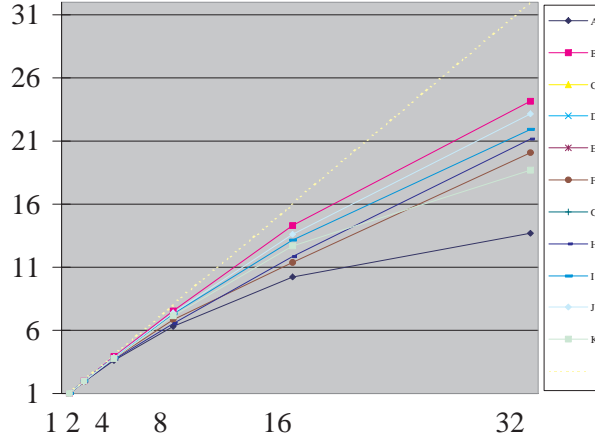
Figure 11: Algorithm I speedups.

Processors	A	B	C	D	E	F	G	H	I	J	K
1	8.0	179.1	292.3	325.2	1072.7	22.7	372.7	70.0	87.3	87.3	2.7
2	4.0	89.4	146.8	163.3	537.9	11.3	187.1	35.0	43.6	43.6	1.4
4	2.0	45.2	73.2	82.4	270.5	5.5	94.3	17.5	21.9	21.9	0.7
8	1.1	22.7	36.8	40.9	136.8	2.8	46.9	8.8	10.9	10.9	0.4
16	0.6	11.4	18.8	20.7	70.0	1.4	24.2	4.4	5.5	5.5	0.2
32	0.3	6.0	10.1	10.8	36.6	0.9	12.4	2.2	2.9	2.9	0.1

Table 6: Algorithm II memory requirements (Mbytes).

Phase	B	C	D	E	F	G	H	I	J
FFT analysis	84.2	2820.5	757.4	299.3	209.9	732.4	128.7	174.6	288.6
2D interpolation	80.3	2001.0	757.6	439.9	172.8	1175.6	108.8	208.8	345.4
Averaging	2.2	3.62	4.0	13.5	0.2	4.6	0.6	1.0	1.0
2D FFTsynth	613.8	1177.0	151.3	1198.7	30.0	1628.5	132.8	44.1	44.1
Exchange	0.8	1.3	1.4	4.9	0.1	1.6	0.4	0.4	0.4
1D FFTsynth	308.4	592.2	72.1	587.7	14.9	818.2	66.6	21.1	21.1
Write output	20.2	33.1	38.4	129.6	3.0	43.6	5.7	10.3	11.9

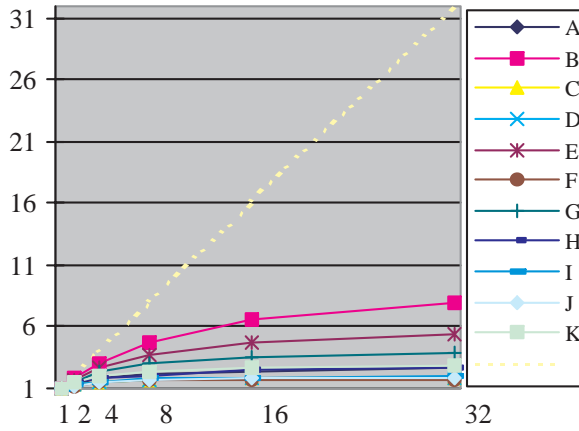Table 7: Time (seconds) for each phase of Algorithm II, one processor.

17

Figure 12: Algorithm II speedups.

Processors	A	B	C	D	E	F	G	H	I	J	K
1	11.4	1018.3	4803.1	1115.3	2194.9	363.6	3380.0	345.0	365.9	558.0	26.4
2	6.7	515.8	2468.5	596.8	1125.5	212.2	1753.0	183.6	206.5	314.7	15.2
4	3.7	261.7	1289.9	319.3	582.4	116.0	914.4	95.0	114.0	175.5	8.3
8	2.5	135.5	718.8	194.2	313.3	72.9	519.9	55.2	68.1	98.7	5.3
16	1.7	69.9	413.0	116.2	162.7	49.5	320.9	34.5	41.5	62.8	3.7
32	0.9	39.3	257.4	71.7	87.5	27.9	166.3	19.2	25.0	36.1	2.5

Table 8: Algorithm III execution time (seconds).

Processors	A	B	C	D	E	F	G	H	I	J	K
1	8.0	179.1	292.3	325.2	1072.7	22.7	372.7	70.0	87.3	87.3	5.2
2	4.6	89.4	146.8	163.3	537.9	11.3	187.1	35.0	43.6	43.6	5.1
4	4.3	45.2	73.2	82.4	270.5	7.8	94.3	17.5	21.9	21.9	5.0
8	4.4	22.7	36.8	40.9	136.8	6.4	46.9	8.8	18.1	18.1	5.5
16	4.9	11.4	18.8	22.8	70.0	6.0	26.1	5.6	17.1	17.1	6.5
32	4.9	11.3	16.1	22.6	52.7	6.6	27.7	4.3	18.8	18.8	8.6

Table 9: Algorithm III memory requirements (Mbytes)

18
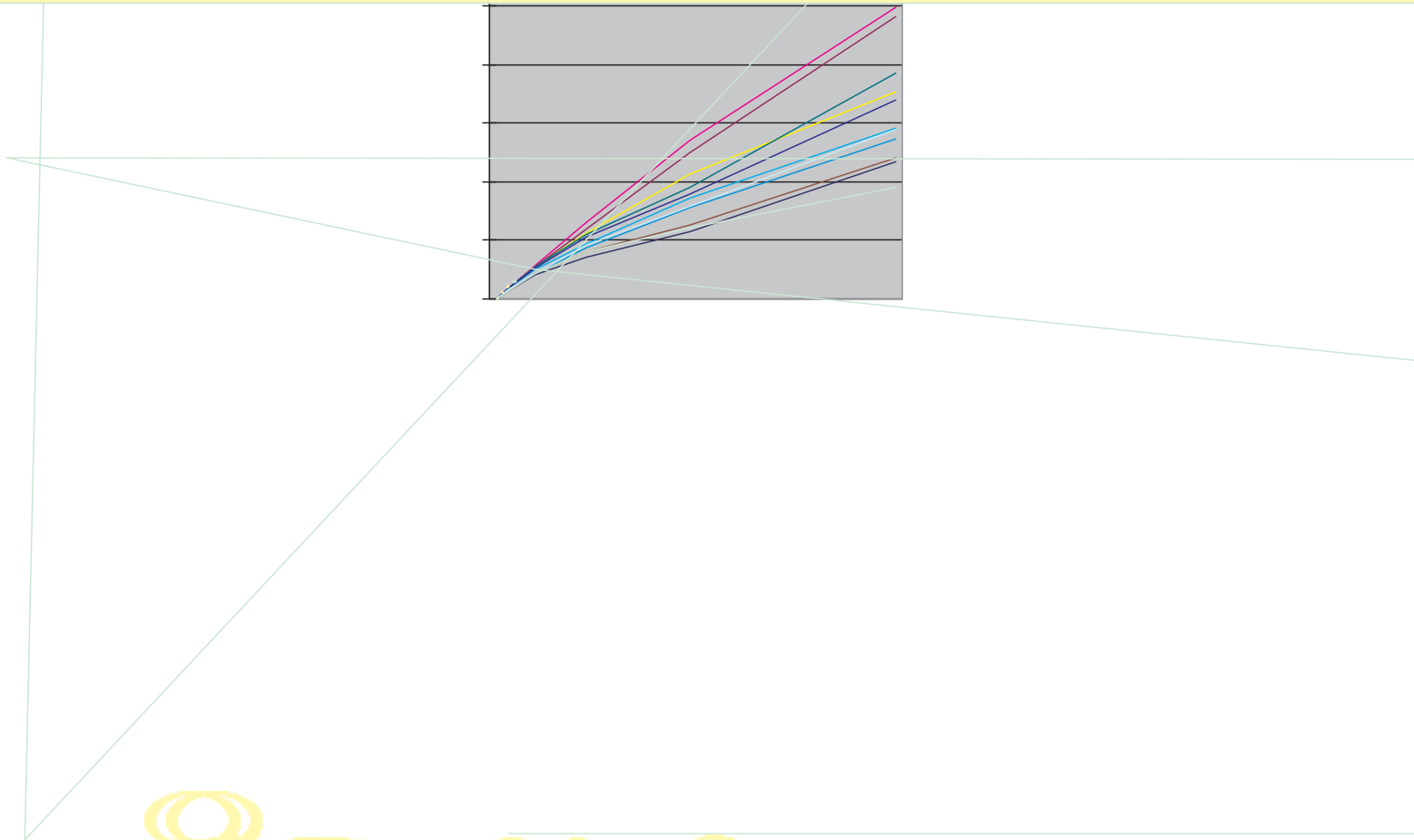
Problem E it went down from about 1.1 Gbytes for one processor to about 36.6 Mbytes for 32 processors. The largest execution time was about 6600 seconds for Problem C with one processor but it decreased to only 3400 for 32 processors. The most computational intensive phases of Algorithm II are the 2D FFT analysis, the 3D FFT synthesis, and the 2D interpolation. The speedups of Algorithm II were abysmal, in the range of 2–8 for 32 processors. For eight out of the eleven structures the speedup was less than 3 on 32 nodes.

Algorithm III seems the most balanced. Indeed its memory requirements scaled down almost linearly with the number of processors and at the same time its speedups reached a maximum value of 25 for 32 processors, for Problem E.

Table 11 shows that the algorithms behave differently relative to one another, for different problems. Algorithm III is constantly better than the others in terms of execution time and speedup with the exception of Problem F where Algorithm I seems to scale slightly better. Algorithm II does not scale well in terms of execution time, it can be much as 12.4 times slower than Algorithm I for execution on 32 nodes.

Two sets of experimental data, Problems I and J, are related to the same virus, Sindbis; for the first we have a set of 389 projections and for the second we have 643, an increase of about 65%. The corresponding increase in the execution time using 32 processors is about 48% for Algorithm I and 44% for Algorithm III. Thus both algorithms scale well in terms of the number of projections. This is very important because the number of projections necessary for the reconstruction of an asymmetric object is almost two orders of magnitude larger than the one for an object with icosahedral symmetry, as discussed in Section 4.

8   Conclusions and Future Work

Data intensive applications require efficient parallel algorithms capable of adapting to the problem size and to the architecture, configuration, and the amount of resources of a target parallel system.

In this paper we present three algorithms for 3D reconstruction of an asymmetric object from its 2D projections, specify their resource requirements, and present results of performance measurements. Our primary interest is in applications to structural biology, in particular to 3D reconstruction of viruses from electron micrographs.

The algorithms we describe have several desirable features. They allow the reconstruction of asymmetric objects, thus they can be used to study the genetic material of the virus core. They scale well in terms of the number of projections used for 3D reconstruction. For more than thirty years the structural biology community has used variations of one of the five algorithms proposed in [Cro70], with the 3D reconstruction being done with Fourier-Bessel transforms. Our algorithms use Cartesian coordinates, are easily parallelizable, and use interpolation instead of least squares as is suggested in [Cro70]. At the time of this writing, we are not aware of any program for the reconstruction of a 3D asymmetric object, of the size and to the resolution we deal with, which is obtained from an arbitrary, sufficiently large, set of the 2D projections of the object.

The number of projections required for the reconstruction of an asymmetric object is almost two orders of magnitude larger than the one for an object with icosahedral symmetry. The structural biologists want to study larger structures including complexes of viruses and antiviral agent, and to increase the resolution of the reconstruction to about 5Å. These factors make a data intensive problem even more demanding in terms of CPU cycles, memory requirements, communication, and I/O bandwidth.

The quality of the solution is the same for the three algorithms outlined in this paper and it is

Problem	Processors	Algorithm II versus Algorithm I		Algorithm III versus Algorithm I	
		Time II/I	Memory II/I	Time III/I	Memory III/I
A	1	0.83	0.69	0.70	0.69
	2	1.19	0.69	0.80	0.92
	4	1.71	0.41	0.82	0.88
	8	2.46	0.22	0.96	0.90
	16	3.56	0.12	1.06	1
	32	4.25	0.06	0.75	1
B	1	0.97	0.67	0.90	0.67
	2	1.07	0.67	0.91	0.67
	4	1.24	0.41	0.91	0.41
	8	1.54	0.21	0.91	0.21
	16	2.11	0.10	0.89	0.10
	32	2.94	0.05	0.84	0.10
C	1	–	–	–	–
	2	1.41	0.67	0.69	0.67
	4	2.32	0.40	0.72	0.40
	8	3.87	0.20	0.74	0.20
	16	7.14	0.10	0.84	0.10
	32	12.38	0.06	0.93	0.09
F	1	0.98	0.83	0.85	0.83
	2	1.56	0.42	0.95	0.42
	4	2.58	0.20	1	0.28
	8	4.17	0.10	1.13	0.22
	16	6.64	0.05	1.29	0.20
	32	11.33	0.03	1.28	0.20
I	1	0.87	0.67	0.70	0.67
	2	1.35	0.67	0.79	0.67
	4	2.16	0.40	0.84	0.40
	8	3.58	0.20	0.96	0.33
	16	5.91	0.10	1.05	0.32
	32	9.48	0.05	1.05	0.35
J	1	0.86	0.67	0.69	0.67
	2	1.36	0.67	0.76	0.67
	4	2.25	0.40	0.83	0.40
	8	3.78	0.20	0.90	0.33
	16	6.42	0.10	1.05	0.32
	32	10.50	0.05	1.03	0.35

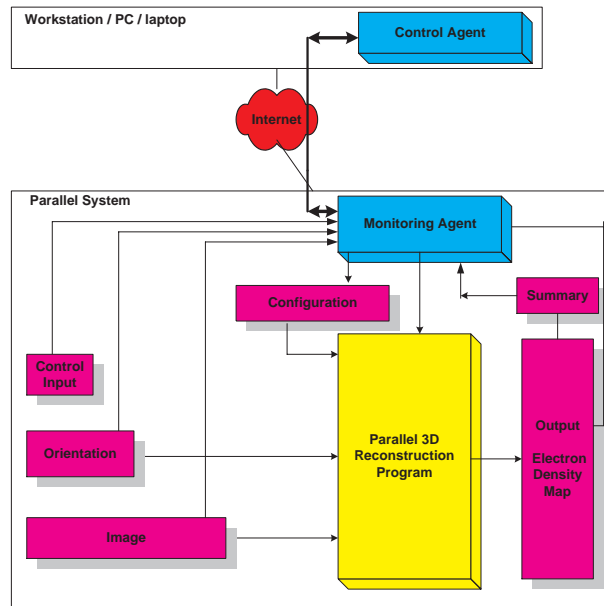Table 11: Execution time and memory of Algorithms II and III compared with Algorithm I

Figure 14: A mixin consisting of agents and the parallel program for 3D reconstruction.

comparable to the one produced by the sequential algorithm described in [Bak98]. All the results we have obtained so far apply only to viruses with either icosahedral or dihedral symmetry.

There is no general method to select the optimal algorithm in a family of algorithms and control its parameters, for a particular problem and a given computing environment without revealing the complexity of the algorithms and of the parallel system to a scientist using the program. We propose to use software agents to facilitate the use of our programs. While it is feasible to use agents for algorithm selection, our first step is to design agents able to control the parameters of the algorithm for a specific problem and computing environment. The structure of the system we are currently implementing is shown in Figure 14.

The Control Agent runs on a workstation, PC, a laptop, or possibly a wireless mobile device connected to the Internet. The function of the Control Agent is to startup a Monitoring Agent on the parallel system used for data analysis. Once started, the Monitoring Agent creates a *Configuration* file using information about the physical problem obtained from the *Control Input* file, the *Image* file, the *Orientation* file, as well as information regarding the configuration of the system gathered by the agent.

The Configuration file describes the number of processors used to run the program, the number of stages of the pipeline of Algorithm III, K-the zero fill aspect ratio.

Once started the parallel 3D reconstruction program runs under the control of the Monitoring Agent. Upon completion, the Monitoring Agent scans the output files of the parallel program and produces a summary of the performance data and sends it back to the Control Agent. It may also send the output data to a graphics device for visualization.

The agents are written using the Blueprint agent description language and run in the Bond environment [BoJ00]. Bond is a Java based agent framework available as a open source under an LGPL license from our web site, **http://bond.cs.purdue.edu**. The system has been used in the past to build a network of PDE solvers, [BoM00], and for several other applications.

9 Acknowledgments

References

[Bak97] Baker, T. S., I. M. B. Martin, and D. C. Marinescu, "A parallel algorithm for determining orientations of biological macromolecules imaged by electron microscopy". CSD-TR #97-055, Department of Computer Sciences, Purdue University, 1997

[Bak98] Baker, T. S., N. H. Olson, and S. D. Fuller, "Adding the Third Dimension to Virus Life Cycles: Three-Dimensional Reconstruction of Icosahedral Viruses from Cryo-Electron Microscopy", Microbiology and Molecular Biology Reviews, Vol. 63, No. 4, 862-922, 1999.

[BoJ00] Bölöni L., K. K. Jun, K. Palacz, R. Sion, and D. C. Marinescu, "The Bond agent system and applications", in *Agent Systems, Mobile Agents, and Applications,* (D. Kotz and F. Mattern, eds.), Lecture Notes on Computer Science, vol. 1882, Springer Verlag, 99–112, 2000.

[BoM00] Bölöni L., D. C. Marinescu, J. R. Rice, P. Tsompanopoulu, and E. A. Vavalis, "Agent-based scientific simulation and modeling", Concurrency Practice and Experience, vol 12, 845-861, 2000.

[Böt97] Böttcher, B., S. A. Wynne, and R. A. Crowther, "Determination of the fold of the core protein of hepatitis B virus by electron cryomicroscopy", Nature (London) 386, 88–91, 1997.

[Bri96] Bricogne, G., Irwin, J., "Maximum-likelihood structure refinement: theory and implementation within BUSTER+TNT", in Molecular Refinement: Proc CCP4 Study Weekend, January 1996, Dodson E., Moore M., Ralph, A., and Bailey S., (Eds). CCLRC Darebury Laboratory, pp. 85-92, 1996.

[Con97] Conway, J. F., N. Cheng, A. Zlomick, P. T. Wingfield, S. J. Stahl, and A. C. Steven, "Visualization of a 4-helix bundle in the hepatitis B virus capsid by cryo-electron microscopy", Nature (London) 386, 91–94, 1997.

[Cro70] Crowther, R. A., D. J. DeRosier, and A. Klug, "The reconstruction of a three-dimensional structure from projections and its application to electron microscopy", Proc. Roy. Soc. Lond. A 317, 319–340, 1970.

[Dea93] Deans, S. R., *The Radon Transform and Some of Its Applications*, 2nd Edit., Krieger Publishing Company, 1993.

[DeR67] DeRosier, D. J., A. Klug "Reconstruction of three-dimensional structures from electron micrographs", *Nature*, 217, pp. 130-134.

[DeR70] DeRosier, D. J., P.B. Moore "Reconstruction of three-dimensional images from electron micrographs of structures with helical symmetry", *J. Molec. Biol*, 52, pp. 355-369.

[Doe00] Doerschuk, P., J. E. Johnson "Ab initio reconstruction and experimental design for cryo electron microscopy", IEEE Trans. on Information Theory, vol. 46, no. 5, pp. 1714-1729, 2000.

[Fra96] Frank, J., *Three-Dimensional Electron Microscopy of Macromolecular Assemblies*, Academic Press, 1996.

[Gor74] Gordon, R., "Three-dimensional reconstruction from projections: A review of algorithms", Intern. Rev. of Cytology 38, 111–151, 1974.

[Gra96] Grangeat, P., and J-L Amans, Eds., *Three-Dimensional Image Reconstruction in Radiology and Nuclear Medicine*, Kluwer Academic Publishers, 1996.

[Har86] Harauz, G., M., B.van Heel, "Exact filters for general geometry three dimensional reconstruction", Optik, 73, pp. 146-156, 1986.

[Joh94] Johnson, C. A., N. I. Weisenfeld, B. L. Trus, J. F. Conway, R. L. Martino, and A. C. Steven, "Orientation determination in the 3D reconstruction of icosahedral viruses using a parallel computer", CS & E, 555–559, 1994.

[Lyn97] Lynch, R. E., and D. C. Marinescu, "Parallel 3D reconstruction of spherical virus particles from digitized images of entire electron micrographs using Cartesian coordinates and Fourier analysis", CSD-TR #97-042, Department of Computer Sciences, Purdue University, 1997.

[Lyn99] Lynch, R. E., D. C. Marinescu, H. Lin, and T. S. Baker, "Parallel algorithms for 3D reconstruction of asymmetric objects from electron micrographs," Proc. IPPS/SPDP, IEEE Press, 632–637, 1999.

[Lyn00] Lynch, R. E., H. Lin, and D. C. Marinescu, "An efficient algorithm for parallel 3D reconstruction of asymmetric objects from electron micrographs," Proc. Euro-Par 2000, Lecture Notes in Computer Science, vol 1900, 481–490, 2000.

[Lyn00a] Lynch, R. E., H. Lin, D. C. Marinescu, and Y. Ji "An Algorithm for parallel 3D reconstruction of asymmetric objects from electron micrographs", (submitted), 2000.

[Mar97] Martin, I. M., D. C. Marinescu, T. S. Baker, and R. E. Lynch, "Identification of spherical particles in digitized images of entire micrographs", J. of Structural Biology, 120, 146–157, 1997.

[Mur97] Murshudov, G.N., A.A. Vagin, E.J. Dodson, "Refinement of macromolecular structures by maximum likelihood method", Acta Cryst. D53, pp. 240-255, 1997

[Pan96] Pannu, N.S., R.J. Read, "Improved structures refinement by maximum likelihood", Acta Cryst. D52, pp. 659-668, 1997

[Rad17] Radon, J., "Über the Bestimmung von Funktionen durch ihre Integralwerte langs gewisser Manningfaltigkeiten" Berichte über die Verhandlungen der Könieglich Sachsischen Gesellshaft der Wissenschaften zu Leiptzig. *Math. Phys. Klasse* 69, 262-277, 1917.

[Ros98] Rossmann, M. G., and Y. Tao, "Cryo-electron-microscopy reconstruction of partially symmetric objects", J. Structural Biology, 125, 196–208, 1999.

[vHe00] van Heel, M., B. Growen, R. Matadeen, E. Orlova, R. Finn, T. Pape, D. Cohen, H. Stark, R. Schmidt, M. Schatz, A. Patwardhan "Single-particle electron cryo-microscopy: towards atomic resolution", Quarterly Review of Biophysics, 33, 4, pp. 307-369, 2000.

[Yan00] Yan X., N. H. Olson, J. L. Van Etten, M. Bergoin, M. G. Rossmann and T. S. Baker, "Structure and assembly of large, lipid-containing, dsDNA viruses", http://expert.cc.purdue.edu/ xdy/research/research.html.